LINK-80 Reference Manual

For Use with the TRS-80
Disk Operating System (TRSDOS)

LINK-80 Linking Loader
Reference Manual

Contents

SECTION 2

LINK-80 Linking Loader

The LINK-80 Linking Loader takes the relocatable object files generated by the FORTRAN compiler and MACRO-80 assembler and loads them into memory in a form that can be executed. In addition, LINK-80 automatically searches the system library (FORLIB) and loads the library routines needed to satisfy any undefined global references (i.e., calls generated by the compiled program to subroutines in the system library).

LINK-80 provides the user with several loading options. Programs may be loaded at user-specified locations, and program areas and data areas may be separated in memory. A memory image of the executable file produced by LINK-80 can be written to disk. The default extension for the name of the executable file is /CMD.

1.1 Running LINK-80

When you give TRSDOS the command

L80

(diskette #2 must be in the disk drive), you are running the LINK-80 linking loader. When the loader is ready to accept commands, it prompts the user with an asterisk. The loader will exit back to TRSDOS after a command containing an E or G switch (see Section 2.1.1), or after a <break> is done at command level.

Command lines are also supported by LINK-80.

1.1.1 LINK-80 Commands

A command to LINK-80 consists of a string of filenames and/or switches. The command format is:

[filename1][-switch1][,filename2][-switch2]...

All filenames must be in TRSDOS filename format.

After LINK-80 receives the command, it will load or search (see the S switch) the specified files. Then it will list all the symbols that remained undefined, with each followed by an asterisk.

*=TEST	Examine file TEST/CRF and generate a cross reference listing file TEST/LST.
*T=TEST	Examine file TEST/CRF and generate a cross reference listing file T/LST.

Cross reference listing files differ from ordinary listing files in that:

1. Each source statement is numbered with a cross reference number.

2. At the end of the listing, variable names appear in alphabetic order along with the numbers of the lines on which they are referenced or defined. Line numbers on which the symbol is defined are flagged with '#'.

Example:

```
*MAIN

 DATA     5200     5300

 SUBR1*       (SUBR1 is undefined)

 DATA     5200     5300

*SUBR1
*-G           (Starts Execution - see below)
```

Typically, to execute a FORTRAN program and subroutines, the user types the list of filenames followed by -G (begin execution). Before execution begins, LINK-80 will always search the system library (FORLIB/REL) to satisfy any unresolved external references. If you wish to first search libraries of your own, append the filenames that are followed by -S to the end of the loader command string.

1.1.2 LINK-80 Switches

A number of switches may be given in the LINK-80 command string to specify actions affecting the loading process. Each switch must be preceded by a dash (-). These switches are:

Switch	Action
R	Reset. Put loader back in its initial state. Use -R if you loaded the wrong file by mistake and want to restart. -R takes effect as soon as it is encountered in a command string.
E or E:Name	Exit LINK-80 and return to the Operating System. The system library will be searched on the current dsk to satisfy any existing undefined globals. The optional form E:Name (where Name is a global symbol previously defined in one of the modules) uses Name for the start address of the program. Use -E to load a program and exit back to the monitor.
G or G:Name	Start execution of the program as soon as the current command line has been interpreted. The system

library will be searched on the current disk to satisfy any existing undefined globals. Before execution actually begins, LINK-80 prints two numbers and a BEGIN EXECUTION message. The two numbers are the start address and the address of the next available byte. The optional form G:Name (where Name is a global symbol previously defined in one of the modules) uses Name for the start address of the program.

N

If a <filename>-N is specified, the program will be saved on disk under the selected name (with a default extension of CMD) when a -E or -G is done.

P and D

-P and -D allow the origin(s) to be set for the next program loaded. -P and -D take effect when seen (not deferred), and they have no effect on programs already loaded. The form is -P:<address> or -D:<address>, where <address> is the desired origin in the current typeout radix. (Default radix is hexadecimal. -O sets radix to octal; -H to hex.) LINK-80 does a default -P:<link origin> (i.e., 5200).

If no -D is given, data areas are loaded before program areas for each module. If a -D is given, all Data and Common areas are loaded starting at the data origin and the program area at the program origin. Example:

```
*-P:200,FOO
Data    200    300
*-R
*-P:200 -D:400,FOO
Data    400    480
Program 200    280
```

U

List the origin and end of the program and data area and all undefined globals as soon as the current command line has been interpreted. The program informa-

tion is only printed if a -D has been done. Otherwise, the program is stored in the data area.

M — List the origin and end of the program and data area, all defined globals and their values, and all undefined globals followed by an asterisk. The program information is only printed if a -D has been done. Otherwise, the program is stored in the data area.

S — Search the filename immediately preceding the -S in the command string to satisfy any undefined globals.

Examples:

```
*-M                                List all globals

*MYPROG,SUBROT,MYLIB-S
                    Load MYPROG.REL and SUBROT.REL and
                     then search MYLIB.REL to satisfy
                     any remaining undefined globals.

*-G                                Begin execution of main program
```

1.2 Sample Link

```
DOS READY
L80
*EXAMPL,EXMPL1-G
DATA    5200    52AC
[5200   52AC]
[BEGIN EXECUTION]

   1792                    14336
  14336                   -16383
 -16383                       14
     14                      112
    112                      896
DOS READY
```

1.3 Format of LINK Compatible Object Files

NOTE

Section 2.3 is reference material for users who wish to know the load format of LINK-80 relocatable object files. Most users will want to skip this section, as it does not contain material necessary to the operation of the package.

LINK-compatible object files consist of a bit stream. Individual fields within the bit stream are not aligned on byte boundaries, except as noted below. Use of a bit stream for relocatable object files keeps the size of object files to a minimum, thereby decreasing the number of disk reads/writes.

There are two basic types of load items: Absolute and Relocatable. The first bit of an item indicates one of these two types. If the first bit is a 0, the following 8 bits are loaded as an absolute byte. If the first bit is a 1, the next 2 bits are used to indicate one of four types of relocatable items:

00 Special LINK item (see below).

01 Program Relative. Load the following 16 bits after adding the current Program base.

10 Data Relative. Load the following 16 bits after adding the current Data base.

11 Common Relative. Load the following 16 bits after adding the current Common base.

Special LINK items consist of the bit stream 100 followed by:

a four-bit control field

an optional A field consisting of a two-bit address type that is the same as the two-bit field above except 00 specifies absolute address

an optional B field consisting

of 3 bits that give a symbol
length and up to 8 bits for
each character of the symbol

A general representation of a special LINK item is:

```
1 00 xxxx  yy nn     zzz + characters of symbol name
           --------  ------------------------------
           A field             B field
```

xxxx	Four-bit control field (0-15 below)
yy	Two-bit address type field
nn	Sixteen-bit value
zzz	Three-bit symbol length field

The following special types have a B-field only:

0	Entry symbol (name for search)
1	Select COMMON block
2	Program name
3	Request library search
4	Reserved for future expansion

The following special LINK items have both an A field and a B field:

5	Define COMMON size
6	Chain external (A is head of address chain, B is name of external symbol)
7	Define entry point (A is address, B is name)
8	Reserved for future expansion

The following special LINK items have an A field only:

9	External + offset. The A value will be added to the two bytes starting at the current location counter immediately before execution.
10	Define size of Data area (A is size)
11	Set loading location counter to A
12	Chain address. A is head of chain, replace all entries in chain with current location counter. The last entry in the chain has an address field of absolute zero.
13	Define program size (A is size)
14	End program (forces to byte boundary)

The following special Link item has neither an A nor a B field:

15	End file

1.4 LINK-80 Error Messages

LINK-80 has the following error messages:

?No Start Address — A -G switch was issued, but no main program had been loaded.

?Loading Error — The last file given for input was not a properly formatted LINK-80 object file.

?Out of Memory — Not enough memory to load program.

?Command Error — Unrecognizable LINK-80 command.

?<file> Not Found — <file>, as given in the command string, did not exist.

%2nd COMMON Larger /XXXXXX/ — The first definition of COMMON block /XXXXXX/ was not the largest definition. Re-order module loading sequence or change COMMON block definitions.

%Mult. Def. Global YYYYYY — More than one definition for the global (internal) symbol YYYYYY was encountered during the loading process.

%Overlaying [Program / Data] Area [,Start = xxxx / ,Public = <symbol name>(xxxx) / ,External = <symbol name>(xxxx)] — A -D or -P will cause already loaded data to be destroyed.

?Intersecting [Program / Data] Area — The program and data area intersect and an address or external chain entry is in this intersection. The final value cannot be converted to a current value since it is in the area intersection.

?Start Symbol - <name> - Undefined

After a -E: or -G: is given, the symbol specified was not defined.

Origin [Above / Below] Loader Memory, Move Anyway (Y or N)?

After a -E or -G was given, either the data or program area has an origin or top which lies outside loader memory (i.e., loader origin to top of memory). If a Y <cr> is given, LINK-80 will move the area and continue. If anything else is given, LINK-80 will exit. In either case, if a -N was given, the image will already have been saved.

?Can't Save Object File

A disk error occurred when the file was being saved.

1.5 Program Break Information

LINK-80 stores the address of the first free location in a global symbol called $MEMRY if that symbol has been defined by a program loaded. $MEMRY is set to the top of the data area +1.

NOTE

If -D is given and the data origin is less than the program area, the user must be sure there is enough room to keep the program from being destroyed. This is particularly true with the disk driver for FORTRAN-80 which uses $MEMRY to allocate disk buffers and FCB's.

EDIT-80 User's Guide

For Use with the TRS-80
Disk Operating System (TRSDOS)

Microsoft EDIT-80 User's Guide

Contents

CHAPTER 1

EDIT-80 Operation

1.1 Introduction

EDIT-80 is a line-oriented and character-oriented text editor. EDIT-80 commands are simple and straightforward, yet powerful enough to accommodate the most demanding user. For the novice or for those requiring only cursory use of EDIT-80, the first four chapters of this document contain all the information necessary to complete a fairly extensive editing session. The remaining chapters describe the enhancements to EDIT-80 that provide the user with more sophisticated techniques.

1.2 Running EDIT-80

To run EDIT-80, type and enter

```
EDIT
```

at TRSDOS command level. EDIT-80 will ask for the filename by typing

```
FILE:
```

Enter the name of your file. Use TRSDOS filename format for the filename:

```
filename[/extension][.password][:drive#]
```

If the filename refers to a file that already exists, type the filename followed by <enter>, and EDIT-80 will read in the file. If the file does not have line numbers, EDIT-80 will append them, beginning with line number 100 and incrementing by 100. After EDIT-80 prints

```
Version x.x
Copyright 1977,78 (c) by Microsoft
Created: xxxx
xxxx Bytes free
*
_
```

it is at commmand level, as indicated by the * prompt. All commands to EDIT-80 are entered after the * prompt.

If the filename refers to a new file to be created, type the filename followed by the <break> key.

EDIT-80 will return the message

```
Creating
Version x.x
Copyright 1977,78 (c) by Microsoft
Created: xxxx
xxxx Bytes free
*
```

Next enter the command I (see Section 2.1 for a further description of the I command). EDIT-80 will type the first line number, 00100, followed by a tab.

```
*I
00100
```

Now you are ready to enter the first line of your file. A line consists of up to 255 characters and is terminated by <enter>. After every line entered, EDIT-80 will type the next line number, incrementing by 100. This is the "permanent increment." (There are various commands that will change the permanent increment - see Chapter 2.) Line numbers 00000 through 99999 are available for use in your EDIT-80 file.

NOTE

Microsoft products such as TRS-80 FORTRAN and MACRO-80 all support input files which include EDIT-80 line numbers.

If a typing error is made while entering or editing a line, use the Delete key (←) to delete the incorrect character(s). If, while typing a line, you wish to erase the entire line and start over, type shift ←.

When you wish to stop entering lines and return to command level, type the <break> key after the next available line number.

1.3 Ending the Editing Session

To exit EDIT-80, enter the Exit command:

```
*E
```

The Exit command writes the edited file to disk under the filename that was used to create the file. Subsequent editing sessions with that file require that a filename be specified with the Exit

command. See Section 6.1.

To exit EDIT-80 without writing the edited file to disk, enter the Quit command:

```
*Q
```

After execution of a Quit command, all the changes entered during the editing session are lost.

1.4 Line Numbers and Ranges

Most commands to EDIT-80 require a reference to a line number or a range of line numbers. A line number is specified by using the number itself (it is not necessary to type the leading zeros), or one of three special characters that EDIT-80 recognizes as line numbers. These special characters are:

. (period)	refers to the current line
^ (up arrow)	refers to the first line
* (asterisk)	refers to the last line

Ranges may be specified in one of two ways:

1. With a colon. The designation

   ```
   200:1000
   ```

   means all lines from line number 200 to line number 1000, inclusive. If lines 200 and 1000 do not exist, the range will begin with the first line number greater than 200 and end with the last line number less than 1000.

2. With an exclamation point. The designation

   ```
   200!3
   ```

   means the range of three lines that starts with line 200. If line 200 does not exist, 200!3 means the range of three lines that starts with the first line after 200.

Here are some examples of line and range specifications (shown here with the Print command):

P.:2000	Prints the range that begins with the current line and ends with line 2000.
P500	Prints line 500.
P.	Prints the current line.

P.!15	Prints the range that begins at the current line and ends after the next 15 lines.
PΛ:1500	Prints the range that begins with the first line and ends with line 1500.
PΛ:*	Prints the entire file.

See Appendix C for more examples of range specification.

1.5 Format Notation

Throughout this document, generalized formats of EDIT-80 commands are given to guide the user. These formats employ the following conventions:

1. Items in square brackets are optional.

2. Items in capital letters must be entered as shown.

3. Items in lower case letters enclosed in angle brackets are to be supplied by the user:

<position>	supply any line number (up to five digits) or ".","Λ" or "*"
<range>	supply any <position> or any <range> <range> = <position>:<position> or <position>!<number>
<inc>	supply a non-zero integer to be used as an increment between line numbers
<filename>	supply any legal TRSDOS filename as described in Section 1.2

4. Punctuation must be included where shown.

5. Items separated by a vertical line are mutually exclusive. Choose one.

6. <break> refers to the break key and is echoed as $. If you see a $ in a format notation, it refers to the break key.

7. In any command format, spaces and tabs are insignificant, except within a line number or a filename.

8. Underlined items are typed by EDIT-80.

CHAPTER 2

Beginning Interline Editing

Editing a file by printing, inserting, deleting and replacing entire lines or groups of lines is termed interline editing. This section describes the commands used to perform these functions.

2.1 Insert Command

The Insert command is used to insert lines of text into the file. EDIT-80 types each line number for you during insert mode. The format of the Insert command is:

I[<position>[,<inc> | ;<inc>]]

Insertion of lines begins at <position> and continues until <break> is typed or until the available space at that point in the file is depleted. (In either case, EDIT-80 returns to command level.)

If no <inc> is included with the command, the default is the permanent increment. ,<inc> specifies a new increment that is then established as the permanent increment. ;<inc> specifies a temporary increment for use with the current command, but does not change the permanent increment.

If no argument is supplied with the Insert command (I<enter>), insertion resumes where the last insert command was terminated, using the last temporary increment. If only <position> is supplied (I<position><enter>), the permanent increment is used.

EDIT-80 will not allow insertion where a line already exists. If <position> is a line number that already exists, the command I<position> will add the permanent increment (or the temporary increment, if one was specified) to <position> and allow insertion at line number <position>+<inc>. If line <position>+<inc> already exists, or if line numbers exist between <position> and <position>+<inc>, an error message will be printed.

The line feed (↓) key may be used to start a new physical line without starting a new logical line, thus providing compatibility with Microsoft BASIC

source files.

Here is an example using the Insert command:

```
*I7740,10
07740           K=K+1
07750           GO TO 400
07760    $
*
-
```

Note that the insertion is terminated with <break>. The <break> key may be typed at the end of the last line inserted (instead of <enter>) or at the beginning of the next line. A line is not saved if <break> is the first key typed on that line.

2.2 Delete Command

The Delete command removes a line or range of lines from the file. The format of the command is:

D<range>

After a Delete command is executed, the current line (".") is set to the first line of the deleted range.

Examples of the Delete Command:

D7000	delete line 7000
D.	delete the current line
D200:900	delete lines 200 through 900
D2000:*	delete all lines from line 2000 through the last line

2.3 Replace Command

The Replace command combines the effects of the Delete and Insert commands. The format of the command is:

R<range>[,<inc> | ;<inc>]

The Replace command deletes all of the lines in <range>, then allows the user to enter new text as if an Insert command had been issued. (EDIT-80 types the line numbers.)

The options for selecting the increment between

line numbers are the same as those for the Insert command (see Section 2.1).

Here is an example using the Replace command:

```
*R500:600;50
00500           DO 80 I=1,7
00550           Y(I)=ALOG(Y(I))
00600    80     CONTINUE
*
```

In the above example, the lines in the range 500 to 600 were deleted and replaced by three new lines (500, 550 and 600), using a temporary increment of 50. Insertion terminated automatically because there was not enough room for EDIT-80 to create line 650.

2.4 Print Command

The Print command prints lines at the terminal. The format of the command is:

P<range>

Examples of the Print command:

P.:700	print all lines from the current line through line 700
P800:*	print all lines from line 800 through the end of the file

Typing <line feed> (↓) at command level will cause the line after the current line to be printed. Typing <break> at command level will cause the line before the current line to be printed. Typing P<enter> will cause the next 20 lines to be printed.

2.5 List Command

The List command

L<range>

is the same as the Print command, except the output goes to the line printer.

2.6 Number Command

The Number command renumbers lines of text. You may wish to renumber lines to "make room" for an insertion, or just to organize the line numbers in a file. The format of the Number command is

N[<start>][,<inc> | ;<inc>][=<range>]

where:

1. <start> is the first number of the new sequence. If <start> is omitted but <range> is included, <start> is set to the first line of <range>. If <start> and <range> are omitted, but <inc> is included, <start> is set to <inc>. If <start> is omitted and <inc> is included and <range> specifies only a page number (e.g., =/2), <start> is also set to <inc> on that page. If <start>, <range> and <inc> are omitted, <start> is set to the permanent increment.

2. <inc> is the increment between line numbers in the new sequence. The options for selection of the increment are the same as those described for the Insert command (see Section 2.1).

3. <range> is the range of line numbers to be renumbered. If <range> is omitted, the entire file is renumbered.

If the current line is renumbered, "." is reset to the same physical line.

If a Number command would result in line numbers being placed out of sequence, or if EDIT-80 cannot fit all the lines using the given increment, an "Out of order" error message is returned.

Due to EDIT-80's internal memory requirements for executing a Number command, an attempt to renumber a very large file may result in an "Insufficient memory" error. If this situation arises, renumber a smaller portion of the file, write it to disk, renumber another portion, and so on. (See Write Command, Section 6.3.)

Examples of the Number command:

N7000;100=200:1000	Lines 200 through 1000 will be renumbered to begin at line 7000 and increment by 100.

N,10=400:*	Lines 400 through the end will be renumbered to begin with 400 and increment by 10.
N9000=10000:*	Using the permanent increment lines 10000 through the end will be renumbered to begin at 9000.
N,100	Renumber the whole file using increment 100.
N,5=2350!10	This command could be used to make room for an insert by compactifying the ten lines starting with 2350.

CHAPTER 3

Intraline Editing - Alter Mode

The interline editing commands discussed thus far let you edit by inserting, deleting or replacing entire lines. Of course many editing situations require changes to an existing line but not necessarily retyping of the line. Editing a line without retyping it is called intraline editing, and it is done in Alter mode.

3.1 Alter Command

The Alter command is used to enter Alter mode. The format of the command is:

A<range>

In Alter mode, EDIT-80 types the line number of the line to be altered and waits for an Alter mode subcommand.

3.2 Alter Mode Subcommands

Alter mode subcommands are used to move the cursor; search for text; or insert, delete or replace text within a line. The subcommands are not echoed on the terminal.

Many of the Alter mode subcommands may be preceded by an integer, causing the command to be executed that number of times. (When no integer is specified, the default is always 1.) In many cases, the entire command may also be prefaced with a minus sign (-) which changes the normal direction of the command's action. For example:

D	deletes the next character
6D	deletes the next 6 characters
-D	deletes the last character
-12D	deletes the last 12 characters

Each Alter mode subcommand is described below. A summary of the subcommands is given in Appendix B.

NOTE

In the following descriptions, $ represents <break>, <ch> represents any character, <text> represents a string of characters of arbitrary length and i represents any integer.

3.3 Cursor Position

The following commands or terminal keys are used to change the position of the cursor in the line. The location of the cursor is called the "current position."

<space> spaces over characters. i<space> moves the cursor i characters to the right. -i<space> moves the cursor i characters to the left. Characters are printed as you space over them.

→ moves the cursor to the end of the line. If preceded by a minus sign, moves the cursor to the beginning of the line.

L prints the remainder of the line and positions the cursor at the beginning of the line. Proceed with the next Alter mode subcommand.

P prints the remainder of the line and recycles the cursor to the current position. Proceed with the next Alter mode subcommand.

W moves to the beginning of the next word. A word is defined as a contiguous collection of letters, numbers, ".", "$", or "%". iW advances the cursor over the next i words. -iW moves the cursor back through i words to the left.

3.4 Insert Text

I inserts text. I<text>$ inserts the given text beginning at the current position. Note that the text must be followed by a <break> or by <enter>.

B inserts spaces (blanks) at the current position. The B command may be preceded by an integer to insert that many spaces. Spaces are inserted to the right of the cursor only.

G inserts characters. iG<ch> inserts i copies of <ch>.

X extends a line. The X subcommand types the remainder of the line, goes into insert mode and lets you insert text at the end of the line. The -X subcommand moves to the beginning of the line and goes into insert mode. (Don't forget to end your insertion with <break> or <enter>.)

3.5 Delete Text

D deletes the character at the current position. iD deletes i characters beginning at the current position. -iD deletes i characters to the left of the current position. Deleted characters are surrounded by double exclamation points.

← The back-arrow key may also be used to delete characters. The character immediately to the left of the current position is deleted. i<back-arrow> is equivalent to -iD.

H deletes (hacks) the remainder of the line to the right of the cursor (or to the left of the cursor if -H is typed) and enters the insert mode. Text insertion proceeds as if an I command had been typed.

K deletes (kills) characters. K<ch> deletes all characters up to but not including <ch>. iK<ch> deletes all characters up to the ith occurrence of <ch>. -iK<ch> deletes all characters up to and including the ith previous occurrence of <ch>. If <ch> is not found, the command is not executed.

O deletes (obliterates) text. O<text>$ deletes all text up to but not including the next occurrence of <text>. iO<text>$ deletes all text up to the ith occurrence of <text>. -iO<text>$ deletes all characters up to and including the ith previous occurrence of <text>.

T deletes (truncates) the remainder of the line to the right of the cursor (or to the left of the cursor if -T is typed) and exits Alter mode.

Z deletes (zaps) words. iZ deletes the next i words. -iZ deletes words to the left of the cursor.

3.6 Replace Text

R replaces text. iR<text>$ deletes the next i characters and replaces them with <text>. -iR<text>$ replaces text to the left of the cursor. The deleted characters are echoed between double exclamation points.

C changes characters one character at a time. C<ch> changes the next character to <ch>. Only the new character is echoed. iC may be followed by i characters to change that many characters; or it may be followed by fewer than i characters and terminated with <break>, in which case the remaining characters will not be changed. -iC does an i<back arrow> and then an iC. The i<back arrow> is echoed between exclamation points.

3.7 Find Text

S searches for a character. S<ch> searches for the next occurrence of <ch> after the current position and positions the cursor before the character. iS<ch> searches for the ith occurrence of <ch>. -S<ch> and -iS<ch> search for the (ith) previous occurrence of <ch> and position the cursor immediately before it. The character at the cursor position is not included in the search. If <ch> is not found, the command is ignored.

F finds text. F<text>$ finds the next occurrence of <text> and positions the cursor at the beginning of the string. iF<text>$ finds the ith occurrence of <text>. -F<text>$ and -iF<text>$ find the (ith) previous occurrence of <text> and position the cursor before it.

3.8 Ending and Restarting Alter Mode

<cr> carriage return. Prints the remainder of the line, enters the changes and concludes altering of that line.

A same as carriage return.

E enters the changes and concludes altering of that line, but does not print the remainder of the line.

N restores the original line (changes are not saved) and either moves to the next line (if an A<range> command is still in progress), or returns to command level.

Q restores the original line (changes are not saved), exits (quits) Alter mode, and returns to command level.

Shift ← Restores the original line, stays in Alter mode and repositions the cursor at the beginning of the line. Echoes as ^Y.

3.9 Extend Command

The Extend command is issued at command level and is used to extend lines. The format of the command is

X<range>

The effect of the X command is equivalent to typing an A command, followed by an X subcommand. After entering an X command, proceed by typing the text to be inserted at the end of the line. Don't forget you are now in Alter mode and may use any of the Alter mode subcommands, once <break> has been typed.

The Extend command is particularly useful for placing comments in assembly language programs.

CHAPTER 4

Find and Substitute Commands

When it is necessary to change a certain portion of text, it is not always immediately known where that text is located in the file. Even with a listing of the file on hand, it is a tiresome task to scan the listing to find the line number of a particular item that must be changed.

The EDIT-80 Find and Substitute commands allow the user to quickly locate text and make necessary changes.

4.1 Find Command

The Find command locates a given string of text in the file and types the line(s) containing that string. The format of the command is:

F[<range>][,<limit>] <enter> | $<string>$

where $ represents the escape key and <limit> is the number of lines containing <string> to be found. A limit of zero will find all occurrences of <string>. The following rules apply to the format of the Find command:

1. If $<string>$ is omitted, the last string given in a Find command is used.

2. If <limit> is omitted and $<string>$ is included, <limit> is assumed to be 1.

3. If <limit> and $<string>$ are omitted, the previous limit is assumed.

4. If <range> is omitted and $<string>$ is included, the entire range from the previous Find command is used.

5. If <range> and $<string>$ are omitted, the search for the previous string continues from the line where the last occurrence was found.

If the search is unsuccessful, an error message is printed.

Here is a sample editing session using Find:

```
*F^:*$WHI(I)$
01100    WHI(I)=0
*F<enter>
01400    IF (P.GT.WHI(I))WHI(I)=P
*A.
01400    .
         .
         .
*F,2$WLO(I)$
01200    WLO(I)=9999
01500    IF (P.LT.WLO(I))WLO(I)=P
*A.
01500    .
         .
         .
*F.:*$AVG$
Search fails
*F$MEAN$
03700    MEAN=SUM/40
*F,0
04200    IF(P.GT.MEAN) M=M+1
06700    WRITE (6,170) MEAN, M
*A4200
04200    .
         .
         .
```

Find the first line that contains WHI(I). Prints line 1100. Find the next one. Prints line 1400. Caught a mistake in this line. Alter it.

Find the first two lines in the file that contain WLO(I) (range is still .:*). Prints lines 1200 and 1500. Alter line 1500.

Find the first line in the file that contains AVG. There aren't any. Try finding MEAN instead. Prints line 3700.
Find all other lines containing MEAN. (Search begins at the line after line 3700.) Finds two more (4200 and 6700). Alter line 4200, etc.

4.2 Substitute Command

The Substitute command locates a given string, replaces it with a new string and types the new line(s). The format of the command is:

S[<range>][,<limit>] <enter> | $<old string>$<new string>$

where $ represents <break>, and <limit> is the number of lines in which <old string> is to be replaced by <new string>. A limit of zero will replace all occurrences of <old string> with <new string>. <new string> may be a null string. The following rules apply to the format of the Substitute command:

1. If $<old string>$<new string>$ are omitted, the strings given in the last Substitute command are used.

2. If <limit> is omitted and $<old string>$<new string>$ are included, <limit> is assumed to be zero.

3. If <limit> and $<old string>$<new string>$ are omitted, the previous limit is assumed.

4. If <range> is omitted and $<old string>$<new string>$ are included, the entire range from the previous Substitute command is used.

5. If <range> and $<old string>$<new string>$ are omitted, substitution continues from where the last substitution left off.

If no occurrence of <old string> is found, an error message is printed.

Example:

```
*SΛ:5000$ALPHA$BETA$
00950   BETA(K)=ABS(1.-LST(K))
01750   WRITE(6,400) BETA(K)
04100   IF (BETA(K).LT.0)GOTO 9000
```

From the first line to line 5000, replace all occurrences of ALPHA with BETA.

CHAPTER 5

Pages

It is possible to divide an EDIT-80 file into sections called pages, which are separated by page marks. The first page of a file is always page 1, and EDIT-80 always enters command level on page 1 of a multiple-page file. Each subsequent page begins with a page mark and is numbered sequentially. On any given page, the complete range of line numbers (00000 to 99999 or any portion thereof) may be used.

If EDIT-80 encounters a form feed while reading in a file, it will enter a page mark at that point in the file. If EDIT-80 encounters a line number that is less than the previous line number, it will automatically insert a page mark so that proper line number sequence may be maintained. When EDIT-80 writes a file out to disk, a form feed is output with each page mark. Then, when the file is listed, each new page of the file starts on a new physical page.

5.1 Specifying Page Numbers

In a single-page file, only a line number is needed to indicate <position>. In a multiple-page file, EDIT-80 must know the page number as well as the line number to determine a <position>. That is, <position> is indicated by

<line>[/<page>]

where

<line> is ".", "^", "*" or a number of up to five digits.

<page> is ".", "^", "*" or a number of up to five digits. When specifying a page, the characters ".", "^" and "*" refer to the current page, the first page and the last page, respectively. If <page> is omitted, the current page is assumed.

Consequently, in a multiple-page file a <range>, which may be indicated by

<position>:<position>
or
<position>!<number>

may also contain page numbers. If the page number is omitted from the first line number in the range, it is assumed to be the current page. If the page

number is omitted from the second line number in the range, it is assumed to be on the same page as the first line number in the range.

Here are some examples of line numbers and ranges that include page number specification:

100/2:*/*	Line 100 on page 2 through the last line on the last page
100/2:*	Line 100 on page 2 through the end of that page
100:*/5	Line 100 on the current page through the last line on page 5
100/*	Line 100 on the last page
100/.:*/3	Line 100 on the current page through the last line on page 3

See Appendix C for more examples of range specification.

5.2 Inserting Page Marks

Page marks may be inserted in the file at the discretion of the user. To insert a page mark, use the Mark command. The format is:

M<position>

The page mark is inserted immediately after <position>. <position> must exist or an error message.will be printed.

The current line reference (".") is retained after a Mark command is executed. That is, if <position> is before ".", then "." will be moved to the next page and will still point to the same physical line.

5.3 Deleting Page Marks

Page marks are deleted with the K (Kill) command. The format of the command is:

K/<page>

The K command deletes the page mark after <page>. For example, in a four-page file, K/2 would delete

the second page mark (the page mark that started page 3), and the pages would then be numbered 1, 2, and 3. The last line number on <page> must be lower than the first line number on <page>+1 before a K/<page> command can be executed.

5.4 Begin Command

Use the Begin command to return to the beginning of a page. The format of the Begin command is:

B[/<page>]

If <page> is omitted, the B command returns to the beginning of page one.

5.5 Other Commands and Page Marks

1. A Delete command that crosses over a page boundary will delete all lines in the range, but will not delete the page mark.

2. A Print command that moves off the current page will print the new page number prior to printing the first line specified in the command.

3. When output is being done with the List command, a form feed will be printed with each page mark, and the page number will be printed on each page.

4. A range specified with an exclamation point may cross a page boundary.

5. If the range specified in a Number command crosses page boundaries, numbering will start over on each new page; the first line number will equal the increment. Consequently, in the Number command, <start> and the first line of <range> must be on the same page.

CHAPTER 6

Exiting EDIT-80

Section 1.3 introduced the Exit and Quit commands for exiting EDIT-80. These two commands will be described more completely in this chapter. An additional command, the Write command, will also be presented.

6.1 Exit Command

The Exit command is used to write the file to disk and return to TRSDOS. The format of the command is:

E[<filename>][-<switch>]

The edited file is saved on the disk under <filename>. When exiting a new file for the first time, <filename> may be omitted. (In which case, the opening filename is assigned.) Otherwise, a new filename is required for each Exit. The previous file serves as a back-up.

The optional <switch> controls the format of the output. (See Section 6.5.)

6.2 Quit Command

The Quit command is used to return to TRSDOS without writing the edited file to disk. To Quit editing, simply enter:

Q

After a Quit command, all changes entered during the editing session are lost.

6.3 Write Command

The Write command writes the edited text to disk and then returns to EDIT-80 command level. It does not exit the editor, and the current position in the file is not changed. The format of the command is:

W[<filename>][-<switch>]

A filename is not required in the first Write of a new file. A filename is required, however, in all subsequent Write and Exit commands.

The optional <switch> controls the format of the output. (See Section 6.5.)

6.4 Index Files

When reading in a file to be edited, EDIT-80 generates information it needs about each block of the disk file. With a small file, this information is generated in a few seconds, each time the file is read in. However, with larger files (5K or more), the time lag required to read in the file becomes significant. Thus, when EDIT-80 saves a file of 42 or more records on the disk, it also saves a small file, separate from the text file, containing the required information about the text file.

This small file is called the index file, and it can be read faster than the text file. EDIT-80 saves the index file under a filename that is the same as the text filename (passwords not included), with a Z preceding the first two letters of the extension. For example, if the file is called FOO/MAC.SAM, the index file is called FOO/ZMA.

When EDIT-80 is asked to edit a file, it first checks for an index file. If an index file exists, EDIT-80 reads the index file instead of the text file. Care must be taken if the text file is modified by another editor or changed and saved in BASIC. The user must then delete the index file prior to editing the text file again with EDIT-80. If the index file is not deleted, EDIT-80 will have meaningless information about the text file.

6.5 Parameters

When reading in a file, EDIT-80 expects it to be in its own representation. If the file appears to be in another representation, EDIT-80 will add line numbers and try to convert the file to EDIT-80 standard format. There are, however, several other representations that EDIT-80 accepts, if the proper switch is appended to the input filename. Switches are always preceded by a dash (-):

filename[/ext][.password][:drive#][-switch]

For example: FOO/BAS.SAM-BASIC

6.5.1 BASIC Switch

If the BASIC switch is appended to the input filename, EDIT-80 will read the file using the following algorithm:

1. All leading spaces and tabs are removed from each line.

2. The first non-blank character must be a digit.

3. From 1 to 5 leading digits are converted to a line number. More than 5 leading digits constitutes a fatal error.

4. A tab is inserted if the first non-digit is not a space or a tab. If the first non-digit is a space, it is replaced by a tab. If the first non-digit is a tab, it is left alone.

5. On output, if UNSEQ (see Section 6.5.2) has been selected, leading zeros in the line number are suppressed and the tab is converted to a space.

Because BASIC uses line numbers to control the sequence of program execution, BASIC users should beware of renumbering with the N command. Microsoft BASIC will ignore page marks from the EDIT-80 file, so a BASIC file may have multiple pages. Insure, however, that no line number appears more than once in the program.

6.5.2 SEQ and UNSEQ Switches

If the SEQ switch is appended to the input filename, EDIT-80 will use the same algorithm to interpret the text file as with the BASIC switch. However, when the file is output, it will be in standard EDIT-80 format, unless the UNSEQ switch is appended to the output filename.

The UNSEQ switch on input tells EDIT-80 to append its own line numbers to the incoming file, regardless of what it looks like. This switch must be used if the incoming file has digits at the beginning of lines with high bits on that are not to be interpreted as line numbers.

On output, the UNSEQ switch must be specified (if it hasn't been already) to output a non-standard file. That is, if BASIC is specified on input and UNSEQ is specified on output, the file will be output in BASIC format. If BASIC was not specified

on input and UNSEQ is specified on output, the file will be output with no line numbers and no trailing tab. If the UNSEQ switch was specified on input and the user wishes to output a standard file, the SEQ switch on output will override the UNSEQ switch.

APPENDIX A

Alphabetic Summary of Commands

Command	Format and Description	Page
Alter	A<range> Enters Alter mode.	15
Begin	B[<page>] Moves to the beginning of <page>. Default is page 1.	25
Delete	D<range> Deletes lines.	11
Exit	E[<filename>][-<switch] Writes the edited text to disk and exits the editor.	6, 26
Find	F[<range>][,<limit>] <enter> | $<string>$ Finds occurrences of <string>.	20
Insert	I[<position>][,<inc> | ;<inc>] Inserts lines beginning at <position> using increment <inc>. With no argument, continues with previous Insert command.	10
Kill	K/<page> Deletes the page mark at the end of <page>.	24
List	L<range> Prints lines at the line printer.	12
Mark	M<position> Inserts a page mark after <position>.	24
Number	N[<start>][,<inc> | ;<inc>][=<range>] Renumbers the lines in <range> so they begin at <start> and increment by <inc>.	13
Print	P[<range>] Prints lines at the terminal. With no argument, prints the next 20 lines.	12
Quit	Q Exits the editor without writing the edited text to disk.	6, 26

Replace	R<range>[,<inc> | ;<inc>] Replaces line(s) using increment <inc>.	18
Substitute	S[<range>][,<limit>]<enter>|$<old string>$<new string>$ Replaces <old string> with <new string>.	22
Write	W[<filename>][-<switch>] Writes the edited text to disk but does not exit the editor.	26
eXtend	X<range> Allows insertion of text at the end of a line.	19

APPENDIX B

Alphabetic Summary of Alter Mode Subcommands

Command	Format	Action
A	A	Prints the remainder of the line, enters the changes and concludes altering of that line
B	[i]B	Inserts spaces
C	[-][i]C<ch>[...<ch>]	Replaces characters
D	[-][i]D	Deletes characters
E	E	Enters the changes and concludes altering of that line
F	[-][i]F$<text>$	Finds <text>
G	[i]G<ch>	Inserts i copies of <ch>
H	[-]H<text>$	Deletes the remainder of the line and enters the insert mode
I	I<text>$	Inserts <text>
K	[-][i]K<ch>	Deletes all characters up to <ch>
L	L	Positions the cursor at the beginning of the line
N	N	Restores the original line and either moves to the next line (if an A<range> command is still in progress) or returns to command level
O	[-][i]O<text>$	Deletes all characters up to <text>
P	P	Recycles the cursor to the current position
Q	Q	Exits Alter mode and restores the original line

R	[-][i]R<text>$	Replaces i characters with <text>
S	[-][i]S<ch>	Finds <ch>
T	[-]T	Deletes the remainder of the line and concludes altering of the line
W	[-][i]W	Moves the cursor over words
X	[-]X	Extends the line
Z	[-][i]Z	Deletes words
[-] →		Moves the cursor to the end of the line
←		Deletes characters
[-][i]<space>		Moves the cursor over characters
<enter>		Prints the remainder of the line, enters changes and concludes altering of that line
Shift ←		Restores the original line, stays in Alter mode and repositions the cursor at the beginning of the line. Echoes as ↑Y.

APPENDIX C

Summary of Notation

The notation used in this document may be defined as follows:

```
<line>     = <number> | . | ^ | *
<page>     = <number> | . | ^ | *
<position> = <line>[/<page>]
<range>    = <position>[:<position> | !<number>]
```

where:

```
<number>   = <digit> | <number><digit>
<digit>    = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Shorthand Notation for Ranges

The following "shorthand" forms of range specifications may be used with EDIT-80 commands.

Shorthand Notation	Equivalent To	Range Specified
/<page>	^/<page>:*/<page>	All of <page>.
/<page1>:/<page2>	^/<page1>:*/<page2>	The first line on <page1> through the last line on <page2>.
:	^/1:*/*	The entire file.
<position>:	<position>:*/*	<position> through the end of the file. e.g., .: is the same as ./.:*/*
:<position>	^/1:<position>	The first line in the file through <position>. e.g., :. is the same as ^/1:./.

APPENDIX D

EDIT-80 Special Characters

<break>	Aborts the command in progress and returns to EDIT-80 command level.
→	Types a tab.
Shift ←	Erases the line being typed and lets you start over. When used in Alter mode, Shift<-- restores the original line, stays in Alter mode and repositions the cursor at the beginning of the line.

Control characters are typed by holding down the shift key, the down-arrow (↓) key and the correct alpha key at the same time.

Control O	Suspends/resumes output (at the terminal or line printer) from an EDIT-80 command.
Control S	Halts/resumes execution of an EDIT-80 command.

APPENDIX E

Error Messages

Fatal Errors

Disk I/O errors are fatal. The corresponding TRSDOS error message will be printed.

Any TRSDOS system error message is fatal.

Illegal line format
Occurs when EDIT-80 finds a line with strange contents or a strange line number. This should not normally occur when editing a file created by EDIT-80. It is usually caused by reading files not meant for editing, such as binary files.

Edit Error Messages

Illegal command
Tells the user a nonexistent or ill-formed command was typed.

Insufficient memory available
Occurs when the user has made enough changes to the file to have exhausted EDIT-80's memory area. This should only happen when a large file has many changes or when large portions of code are being inserted or renumbered. A W command should be done to compress memory.

No string given
Tells the user the F or S command was given without a search string. This usually happens when using the F or S command with no arguments prior to issuing an F or S command with arguments, or when an <escape> without a search string is typed following the range.

No such line(s)
This message is issued if a command references a line or range which does not exist. Usually occurs when the proper page number is omitted from the line or range.

Line too long
This message is issued when the user attempts to enter a line longer than 255 characters. This may happen when the line is read or as a result of a command which alters the line.

Out of order
Indicates that the line numbers in the file would not be in ascending order if the command were to be executed. This frequently happens when trying to insert where there is not

enough room or trying to delete a page mark.

Search fails
An informative message that tells the user a search was unsuccessful.

Wrap around
This message is printed whenever a line greater than 99999 would be generated.

File Errors

File already exists
Issued if the user tries to give the name of an existing file to a new file, or tries to rename a file using the name of an existing file in an E or W command.

File not found
Issued if the file specified in a command could not be found.

Illegal file specification
Informs the user that the command string contains an illegal character of some kind.

APPENDIX F

Output File Format

Compilers and assemblers should ignore the line numbers and page marks included in EDIT-80 output files (except when included in listing files). Microsoft TRS-80 FORTRAN and MACRO-80 both do so.

A line number consists of five decimal digits followed by a tab character. All six bytes have the high order bit (bit 7) equal to one. It is not recommended that EDIT-80 files be listed with the TRSDOS LIST command. Graphics characters may appear in the line numbers. Use EDIT-80's Print command instead.

When writing a file with -BASIC set, the line numbers have the high order bits equal to zero. Each line number is followed by a space that has the high order bit equal to zero.

A page mark is a form feed character with the high order bit equal to one.

Index